# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

**4,800**
Open access books available

**122,000**
International authors and editors

**135M**
Downloads

Our authors are among the

**154**
Countries delivered to

**TOP 1%**
most cited scientists

**12.2%**
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

8

# Developing Attention-Aware and Context-Aware User Interfaces on Handheld Devices

Massimo Ancona[1], Betty Bronzini[1], Davide Conte[2] and Gianluca Quercini[3]
[1]*University of Genoa, Department of Computer Science, Genoa*
[2]*Eurocontrol SpA, Genoa*
[3]*University of Maryland, Institute for Advanced Computer Studies, College Park, MD*
[1,2]*Italy*
[3]*USA*

## 1. Introduction

*I do not fear computers. I fear lack of them.* - Isaac Asimov

In today's modern societies the lack of computers feared by Asimov is not an imminent danger thanks to the advance of mobile technology in the last two decades. According to several market surveys, sales of handheld devices, especially smartphones, are growing at an incredibly fast rate and are expected to exceed those of any other electronic device by the end of 2011 [1]. This success is far from surprising, as today's handheld devices feature high computational power and provide a wide range of applications that go beyond the traditional use of a phone. Examples are *mHealth*, a term coined by Istepanian et al. (2005) that refers to the use of mobile applications in healthcare, and *augmented reality*, defined by Azuma (1997) as a variation of *virtual reality* that "allows the user to see the real world, with virtual objects superimposed upon or composited with the real world". Essentially, handheld devices fit in a pocket and provide most of the functionalities of a bulky computer. Their small size, however, is a mixed blessing, as it imposes serious limitations on usability, which is the focus of this chapter. In particular, we discuss two important aspects of the interaction with handheld devices, namely *context-awareness* and *text entry*.

*Context-aware computing*, introduced by Schilit et al. (1994), is a major achievement in mobile computing. In fact, unlike desktop computers, handheld devices work in dynamic environments, where user's context and/or location is likely to rapidly change. Based on the information derived from the context, applications can automatically change their interface or behaviour accordingly. For instance, the context-aware application described by Ancona et al. (2006) senses that a certain area of an archaeological site is crowded at a certain time of the day and suggests the visitor to modify his/her visit path. Similarly, the location-aware application presented by Ancona et al. (2003; 2001) understands which ward a doctor is walking in and proposes an appropriate list of drugs that can be rapidly selected to fill in a prescription. Essentially, context-aware applications minimize or remove, if at all possible, the need to interact with the device, by automatically updating its interface and behaviour.

[1] `http://www.gartner.com/it/page.jsp?id=1550814`

Context-awareness can also help to ease tasks that demand a continuous attention of the users, such as *text entry*. Since most handheld devices do not have a hardware keyboard, which would limit their portability, text entry is usually performed by using either a virtual keyboard or a handwriting recognition system. However, neither the former nor the latter stands out in the same way as the *Qwerty* keyboard does for desktop computers.

The remainder of the chapter is organized as follows. In Section 2 we describe the main features of today's smartphones, which are the most popular handheld devices on the market. In Section 3 we discuss issues related to the design of context aware applications, along with examples drawn from our own past research projects *Agamemnon*, *Past* and *WardInHand*. We note that when we worked on those projects, smartphones were not equipped with the sophisticated sensors they feature today. Consequently, the development of context-aware applications was much more challenging than today, as we detail in Section 4. In Section 5 we analyse the problem of text entry and describe a context-aware text entry tool named *WtX* that we developed few years ago. Finally, Section 6 concludes the presentation.

## 2. Smartphones: an overview

A *smartphone* is a cellular phone integrating standard mobile phone capabilities with the advanced computing ability and peripherals of a personal digital assistant (PDA), like a camera, an advanced (touch)screen, a (virtual) keyboard and a compass. Users can interact with the touchscreen either through their fingers or a tiny pen, generally referred to as a *stylus*; the use of a smartphone with a stylus is generally referred to as *pen computing*. Like PDAs, smartphones offer a full featured operating system and an advanced platform for application development.

Although the term *smartphone* was used for the first time in 1997, when Ericsson launched the concept phone GS88, it came into use later in early 2000. Since their early commercial introduction (end 90s-early 2000), smartphones have evolved at an incredible fast pace. In the design phase of our project *Agamemnon*, started on January 2004, we envisioned to use the *Nokia 6600* and we finally resorted to its successor, the *Nokia 6630*, for the implementation phase. After only 7 years, both devices are obsolete and appear like archaeological finds compared to the *Apple iPhone 4*, the *Samsung Galaxy S2 Smartphone* and the *Nokia 3720 Classic*, which are the best selling smarphones on the market at the time we are writing. IMS research states that up to 420 million smartphones (28% of the mobile phone market) will be sold by the end of 2011 and predicts that annual sales will surpass one billion devices (50% of the mobile phone market) by the end of 2016 [2]. According to Gartner, in the first quarter of 2011 *Nokia* is still the worldwide leader of the smartphone market, although its market share considerably declined in favour to *Samsung* [3]. Although *Apple* market share is only 3.9%, it doubled the sales of *iPhones* since 2010. Still referring to the first quarter of 2011, *Android* and *Symbian* dominate the operating system market, but *Apple iOS* is keeping the pace.

A particularly popular smartphone is the *camera phone*, which can be defined as a smartphone with limited capabilities featuring a medium/high resolution camera. According to the latest

[2] http://imsresearch.com/press-release/Global_Smartphones_Sales_Will_Top_420_Million_Devices_in_2011_Taking_28_Percent_of_all_Handsets_According_to_IMS_Research

[3] http://www.gartner.com/it/page.jsp?id=1689814

report by *Strategy Analytics* [4], worldwide camera phone sales will exceed 1 billion units in 2011. The fastest growing segment will be the high-tier camera phone market with sensors of eight megapixels and above.

Clearly, the evolution of smartphones is primarily due to electronic miniaturization that allows powerful processors in small devices. The computing power of smartphones is almost comparable to that of computers that were sold few years ago, which paves the way to applications that were virtually unthinkable before. Consequently, smartphones and PDAs are not only passive devices waiting for any input to perform an action, but they actively support users in such activities as visiting a city or monitoring their health.

### 2.1 Sensors

Sensors can be either *mobile*, if they are installed on mobile devices, or *fixed*, if they are stationary; mobile devices can typically access fixed sensors via a communication network such as *UMTS*, *GPRS*, *WiFi* or *Bluetooth*. Today's smartphones feature a number of different sensors, including GPS receivers, accelerometers, gyroscopes, digital compasses, proximity and ambient light sensors. The *iPhone* features a relatively new sensor, named *Asahi Kasei's azimuth magnetometer*, which determines the orientation of the phone.

The role of sensors is crucial in mobile computing, because, unlike desktop computers, handheld devices are meant to support users in their interaction with the external environment besides the "virtual world" trapped inside the machine. Not surprisingly, sensors greatly simplify the design of context and location aware applications. For instance, by combining information from a GPS receiver and a magnetometer, an application can easily determine not only the position of the user, but also what s/he is currently looking at.

Interestingly, most of the peripherals of handheld devices, such as a camera, are meant to point to the real world, as opposed to peripherals of desktop computers, such as a mouse. For example, *TinyMotion*, described by Wang et al. (2006), is a software that detects in real time a person's hands movements by using the camera of a mobile phone. Similarly, Ancona et al. (2007) described a system, of which an overview is presented in Section 4, that uses the camera of a smartphone and image recognition techniques to determine the position and the orientation of a person. In the first case, the camera is used as an accelerometer, while in the second as a GPS receiver and a magnetometer. We note that it is not appropriate at all to define the camera as a sensor, but many applications use it as such, although from the users' perspective it is merely a tool to take photos.

The camera is not the only example of a device that can be used as a sensor even if it is not its primary purpose. In a cellular network the position of a mobile phone can be estimated based on the knowledge of the *base station* to which the phone is connected. In a sense, the *base station* is used as a location sensor. The location accuracy is about 1km (0.6 mi) in cellular networks based on *microcells*, such as *GPRS*, and only 100m (328 feet) in networks based on *picocells*, such as *UMTS*. These location methods cannot achieve the accuracy of GPS and are typically used when a device does not have a GPS receiver, which was the case of most smartphones and PDAs few years ago, or GPS is not available.

A complete discussion about the role of sensors is outside the scope of this presentation. For further details, we refer the interested reader to the paper written by Lane et al. (2010).

[4] http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=6216

## 3. Context aware applications

As already noted by Turk & Robertson (2000), human-computer interaction is still dominated by the "typing, pointing and clicking" paradigm, which works well with desktop or laptop computers, but reveals all its limitations with handheld devices, due to their limited size.

One way to ease the interaction with a handheld device is to design *context-aware applications*, which "adapt according to the location of use, the collection of nearby people, hosts and accessible devices as well as to changes to such things over time", as defined by Schilit et al. (1994). Context-awareness proved to be particularly effective in healthcare, as witnessed by the wealth of projects and applications that have been proposed in the last decade, of which a good review has been written by Bricon-Soufand & Newman (2007). The ability of an application to automatically change its behaviour based on the context is particularly helpful in handheld devices, as they reduce the number of interactions with the device. To this extent, several models of user interfaces have been introduced, in particular:

- **Perceptual interfaces**, described by Turk & Robertson (2000) as being able to add human-like perceptual capabilities to an application. For instance, an application may be aware of what the user is saying or what the user's face, body, and hands are doing.
- **Computer vision-based interfaces**, presented by Turk (2004) as those that "look at people" and aim at determining the user's focus of attention based on visual clues.
- **Attentive interfaces**, that, as pointed out by Vertegaal (2003), aim at understanding what the user might be interested in at a given time and context and generate relevant information accordingly.
- **Multimodal interfaces**, that integrate, in an homogeneous way, different human communication mechanisms like speech, gestures and eyes and body movements. An overview is presented by Oviatt & Cohen (2000).

Most of the applications we developed are based on attentive interfaces. For instance, our text entry tool *WtX* (Section 5.4) only suggests words that are appropriate to the user's context; the *Agamemnon* system (Section 3.3) guides tourists through an archaeological site by showing on the screen of their smartphones only the information relevant to the monuments which they are currently looking at.

### 3.1 Design principles of a context aware HCI

The design of mobile applications is significantly different from that of desktop applications, which can rely on powerful devices connected to a power outlet and equipped with keyboard, mouse, plenty of storage space and a wide screen to visualize as much information as needed. As opposed to that, handheld devices have a limited autonomy, they lack effective text entry tools and their screen size is typically very small, which poses serious challenges to applications; the *iPhone* 4 screen, for example, is only 3.5 inches. As a result, applications need to visualize as little information as possible in order to avoid visual clutter. On the one hand, providing all information in just one page is not an option, as users would be forced to use such graphical widgets as scrollbars, which are very difficult to handle in small devices. On the other hand, partitioning the information into many pages could make the navigation hard and force users to continuously switch pages. A feasible solution consists of a compromise between the two approaches: grouping only the most significant data in a page while showing the rest only upon users' request.

Here context and location awareness can contribute to simplify the HCI of mobile applications. Consider, for example, the case of an application on a PDA that supports doctors and nurses while filling in prescriptions for their patients in a hospital. In order to optimize the space on the screen, the application only visualizes information about patients in the ward in which the doctor or the nurse is, while other data, if needed, can be visualized through a limited number of taps. Or, the application may be smart enough to understand which ward the doctor is in and automatically update the information on the screen without the explicit request of the doctor.

In this section we show how context and location awareness can be integrated in the design of the HCI. We say that a component of the interface is *horizontal* if it provides utilities that are likely to be shared by several applications. Examples of horizontal components are virtual keyboards, Web browsers, word processors and editors. Similarly, a *vertical* component is one that is linked to a specific application. Here we give to the HCI a very wide meaning by including in it any tool supporting data I/O between a user and a software application or component. This distinction is useful for designing a Service Oriented Architecture (SOA) where the horizontal components identify services i.e. units of computer work, while vertical components identify modules or objects inside a single service.

Today, in the development of the HCI for a wide range of applications, including industrial control systems, power system automation and telemedicine, there is the tendency to shift from ad-hoc solutions to Web services. This is motivated by the wide portability of web browsers that offer almost the same interface on every system - from powerful servers to small handheld devices. Major advantages of web browsers are:

- *No cost*, due to the availability of free software available on all platforms;
- *Intuitiveness*, as web browsers are well-known by any computer user and thus can be considered as a universal interface;
- *Scalability and portability*, as browsers are available for every architecture, from supercomputers to desktop PCs and handheld devices.
- *Open architecture*, as several open and standard systems are available (e.g., Javascript, Java, HTML, XML etc.);
- *Reusability*, as browsers reduce the risks of obsolescence.

An example of Web-based application is *WardInHand*, described in greater details in Section 3.2, that runs under *Windows* and *Linux* and on several devices, from palmtops (small laptops), to full featured notebooks. We successfully experimented it on a *iPAQ* running *Familiar Linux* with the *Konqueror* Web browser. By enhancing the use of horizontal interfaces like web browsers, supported by open software architectures, and by extending the capabilities of existing ones (see *WtX* described later), the application presents an easy-to-learn standard interface with almost the same look-and-feel on every kind of available device. As a result, the software is scalable and reusable and its implementation is exposed to a limited risk of obsolescence, as most of its (horizontal) components are regularly updated.

### 3.2 *WardInHand*

*WardInHand* (abbreviated as WIH) was a three-year EU co-funded IST project (IST 10479, 2000-2002), whose objective was to support the daily activities of doctors and nurses in a hospital. WIH provides a tool for workgroup collaboration and ubiquitous access to the patient's clinical records at every point-of-care through the use of HP iPAQ pocket PCs.

The project has been completed at the end of March 2002 with the release of a system prototype improved with all suggestions and feedbacks collected during the testing phase and evaluation activities extensively performed by the three end users.

WIH belongs to a relatively new class of "everyday and everywhere" applications. In 1999-2000, wireless networks and handheld devices were a completely new technology, making data ubiquity in hospitals a reality, which saves doctors' time and makes handheld devices part of the usual doctors' equipment. On the downside, handheld devices have limited autonomy, especially when they use intensively a wireless network to send and receive data, they provide a small screen real estate and, as already explained, they are hard to interact with. In general, we noticed that the short battery life did not pose any serious problem, mostly because the personnel of the hospital needs WIH only for short time intervals, namely when visiting patients and prescribing treatments.

In order to improve the interaction with the system in uncomfortable situations, we tested some natural interfaces, which, as explained by Abowd & Mynatt (2000), "facilitate a richer variety of communications capabilities between humans and computation". For example, we tested speech recognition technologies as well as handwriting recognition systems, which require less attention from the user than a virtual keyboard to enter a text. We based our HCI on Web technologies integrated with two small components dedicated respectively to textual data input and speech recognition.

The WIH system is composed of a server that hosts a *MySQL* database storing all patients' records and a *Patient Record Manager* (PRM), a Java servlet application managing the clinical data. The PRM interface supports three pen-based mechanisms for data input: icon clicking, selection of items in (pop-up) menus or lists and alphanumeric data input in forms, by using either a virtual keyboard or a handwriting recognition system or *WtX* (Section 5.4). The patient record is composed of six sections, personal info, physiological signs, treatments, tests, annotations and historical info; each is represented as an icon in the application. The interface of PRM visualizes a weekly summary of each physiological sign and treatment, that gives a comprehensive overview of the patient's current health status. A daily overview is also provided, detailing all physiological signs occurred in the last 24 hours. Finally, an interface is provided to update the patient's record with new information, such as new physiological signs and/or prescriptions. We note that the interface of the PRM makes an heavy use of menus and lists, which reduces the need of typing long words.

Further details about WIH are provided by Ancona, Dodero, Minuto, Guida & Gianuzzi (2000).

### 3.3 *Agamemnon* and *Past*

*Agamemnon* (IST-508013) [5], described in details by Ancona et al. (2006), was a research project co-funded under the 6th Framework Program of the European Commission (January 2004 - June 2006), aiming at the development of a dynamic electronic tourist guide to support visits to archaeological sites. It has been successfully tested in two of the most important archaeological sites in the Mediterranean area, namely Paestum (Italy) and Mycenae (Greece). *Agamemnon* is a software system based on a *client-server* architecture. The client is a a *Java* application, running on smartphones with *Symbian* OS 6, that drives the user through the site by displaying information about the monuments being visited; the server is a set of

[5] `http://services.txt.it/agamemnon/`

applications running on workstations, that manage all the cultural data about the site and the requests from the clients. Clients communicate with the server applications via the UMTS network. Before the visit, the user needs to install the application on her phone and fill in a form to let the system know some personal data and preferences, such as her age, cultural background, cultural preferences and available data. Based on the user profile, the *Visitor Profiler*, a software module of the server, creates a personalized path through the site; if the user changes her path, the system modifies the initial path accordingly, adapting it to the user's current choices. The initial path may also be changed autonomously by the system to delay the visit to certain monuments if they are too crowded at a given time.

One of the most interesting and novel aspects of *Agamemnon* is the possibility for the user to take a photo at a monument with the camera integrated in the smartphone and have the system recognize the monument to get additional information about it. The *Agamemnon* image recognition system, described in greater details by Pittore et al. (2005), is based on statistical pattern recognition and realized by using multi-class *support vector machines*. The system is able to recognize not only a monument, but also a detail of it, such as a column, which is useful when the user is not only interested in a generic description of the monument, but wants also to know details about its architecture.

We note that *Agamemnon* represents an evolution over a traditional audioguide that is often rented to visitors in museums and archaeological sites. First of all, an audioguide does not suggest any personalized path through the site, as *Agamemnon* does. Moreover, audioguides are expensive for both the site and the visitors, who usually need to pay an extra cost to rent it; as opposed to that, the *Agamemnon* client application can be freely downloaded from the Internet and installed on the phone of the visitor. The only cost comes from the use of the UMTS network. At the time we developed *Agamemnon* the use of wireless networks was not so widespread as today, especially in the countries where we tested the application, which explains why we resorted to the UMTS network. However, today it is not unrealistic to assume that an archaeological site has a wireless network that visitors can use with little or no cost at all.

*Agamemnon* is the natural evolution of a previous project named *Past* (IST-1999-20805) [6] that aimed at supporting context and location aware visits to archaeological sites. Similarly to *Agamemnon*, the *PAST* consortium included three archaeological sites, namely *Bibracte* (France), *Toumba* (Greece) and *Passo di Corvo* (Italy). Visitors are given a PDA, which in our case was an *HP iPAQ*, with an application that guides them through the site. The application communicates via a wireless *IEEE 802.11b* network with a server application that tracks the position of the visitors at any time to display content relevant to what they are currently seeing. As in *Agamemnon*, the application suggests a visit path based on personal preferences. We explored different technologies to determine the location of a visitor, namely GPS, triangulation of the distances between the visitor and multiple WiFi access points and comparison between images taken by the PDA's webcam and known profiles. One negative aspect of *PAST* is the use of ad-hoc devices, such as PDAs, owned by the archaeological site, not by visitors, as in the case of *Agamemnon*. As a result, the archaeological site must bear the cost of renting devices. Further details on *Past* are provided by Ancona, Dodero, Gianuzzi, Bocchini, Vezzoso, Traverso & Antonacci (2000).

[6] `http://www.beta80.it/past/index.htm`

## 4. *Agamemnon in City*: Context-awareness without sensors

One of the weak points of *Agamemnon* is the need of installing a full application on the visitor's mobile phone, with all that implies in terms of compatibility, software maintenance and user assistance. For this reason, we implemented and successfully tested an extension to *Agamemnon* that only requires a MMS enabled camera phone to support users in their visit to a city. More specifically, the user can take a photo at a monument and submit it via a MMS message to a predefined server application, which recognizes the monument and sends back some information about it, as shown in Figure 1. In some sense, the camera is used as a

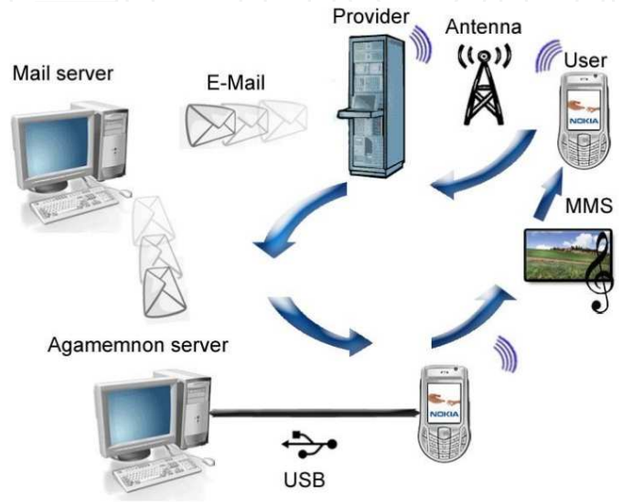



Fig. 1. Architecture of *Agamemnon in City*.

"virtual eye" of the system that tracks what the user is interested in or, in other words, his/her *focus of attention*. We note that we developed this prototype in 2007, when smartphones were not equipped with any sophisticated sensor. In our prototype, the camera itself is used as a sensor to understand both the exact location and the orientation of the user at the same time; in other words, the camera is used as a GPS receiver and a magnetometer, which was the main novel aspect of our prototype. During a cultural visit to a city people are constantly on the move, going in and out of buildings and stopping to admire the main landmarks. While moving people may see something that captures their attention and want to know immediately what it is. Our prototype *Agamemnon in City* aims at satisfying this need and in a sense it can be considered as the natural evolution of a traditional audioguide.

The *Agamemnon in City* prototype is based on the *Agamemnon* image recognition system, which proved to be effective in recognizing monuments of two archaeological sites, namely Paestum and Mycenae. Understanding the user focus of attention from a photo is an intriguing goal with a lot of issues that need to be taken care of. Recognizing objects in a photo is a hard problem for which no system is known to achieve a 100% accuracy. Recognizing monuments is even more challenging, due to the visual similarities shared by landmarks of the same type, such as statues. As a result, the system may not be able to send any information in response to a MMS, which is likely to irritate the sender. Our prototype uses techniques, explained in Section 4.2, to understand the approximate location of the user so as to send, in case a photo cannot be recognized, a list of monuments that are nearby.

### 4.1 Heavy-client *Vs.* light-client

*Agamemnon* is a *heavy-client* system, in which the client application provides a rich interface and advanced features. In our view, the need of installing an application on the visitor's phone is the inherent weakness of the *heavy-client* approach, as the application may not be compatible with all phones and multiple versions need to be developed to run on different platforms and operating systems. In other words, the application implies a cost, which depends on its complexity. In the specific case of *Agamemnon*, we also found some problems due to the choice of *Java* as the main programming language used to develop the client application. *Java*, in fact, imposes some limits on handling high-resolution images, which was a serious problem for the *Agamemnon* image recognition system. Finally, we found that the client application in *Agamemnon* was too demanding in terms of the interaction requested to the user. Therefore, users would spend more time looking at the phone screen than enjoying their visit. In other words, *Agamemnon* does not provide a "eyes-free user interface", in the sense explained by Raman & Chen (2008). These considerations are the rationale for our choice of developing *Agamemnon in City* as a *thin-client system*, where most of the computation is demanded to the server application. In *Agamemnon in City* the phone only needs to be equipped with a camera and to be able to send MMS messages, two requirements that are met by virtually all mobile phones on the market today.

### 4.2 Geolocation

At the time we developed *Agamemnon in City*, it was quite clear that soon most of the smartphones would feature at least a GPS receiver, although the first release of the *iPhone*, dating back to January 2007, did not include one. Therefore, we were interested to see how the knowledge of the user location improves the recognition of the monuments in the photos. To this extent, we modified the prototype to allow users to specify their coordinates while sending a photo to the server application. The server application uses the geographic coordinates to help the image recognition system to correctly interpret the monument in the photo. For example, if the system selects two different monuments as possible interpretations, the one that is far from the user's position is discarded.


Fig. 2. Mapping UMTS cells to landmarks in Genoa, Italy.

Since the mobile phones used for *Agamemnon in City* had no GPS receiver, we used en external one that we interfaced with the phone via *Bluetooth*. The GPS coordinates could be read on the screen of the phone, so that the user could include them manually in the MMS. We note that with today's smartphones, the geographic coordinates would be automatically integrated in the photo, so there would be no need to specify them manually. One of the problem of the GPS receiver we used is that it did not receive properly the signal from the satellites in the narrow streets of the old town of Genoa (Italy), the city where we tested the prototype.

For this reason, we also used another geolocation technique termed *Cell-ID*, of which an interesting discussion is provided by Warrior et al. (2003). Basically, *Cell-ID* consists in locating a phone based on the cell to which the phone is connected. The location accuracy is acceptably good, as the radius of the area covered by a UMTS cell is approximately 100 meters. By using a free software, we mapped the main landmarks of Genoa to the identifier of the UMTS cells covering the area where they are, as shown in Figure 2. We note that by using *Cell-ID* we do not need any additional hardware, but we need a simple software that shows on screen the identifier of the cell to which the phone is connected. Obviously, *Cell-ID* is suitable in cities, where landmarks are usually spatially dispersed, but is useless in the archaeological sites such as the ones in Paestum and Mycenae, because there may be more than one monument within 100m.

There are several other geolocation methods, of which an overview is presented in Table 1. *GPS*, *Cell-ID* as well as *DGPS* and *WAAS*, which are enhancements of *GPS*, are *global*

| **Technology** | **Accuracy (m)** | **Cost** | **Coverage** | **Pros** | **Cons** |
|---|---|---|---|---|---|
| GPS | 10 | High * | Unlimited | Accurate | Costly * |
| Cell-ID | 100 | None | Unlimited | No hardware | Unaccurate |
| DGPS | 0.1 | High | Unlimited | Very accurate | Costly |
| WAAS | 2 | Very high | Unlimited | Accurate | Costly |
| Bluetooth | 20 | Low | Indoors | Popular | Limited range |
| WiFi | 20 | High | Indoors | Fast | Complex |
| RFID | 0.01 | Low | Indoors | Precise | Limited range |

* Referred to when we developed the prototype

Table 1. Comparison of geolocation methods.

*positioning systems*, as they provide a global coverage and they are indeed used to track devices outdoors. In particular, *DGPS* and *WAAS* are very accurate but their cost prevents their use in applications such as our prototype *Agamemnon in City*. On the other hand, *Bluetooth*, *WiFi* and *RFID* are *local positioning systems* that use a set of beacons that only detect devices that are nearby. Since *Bluetooth* is provided in the vast majority of mobile devices, including the old ones, it is more cost-effective than both *WiFi* and *RFID*.

As a side project of *Agamemnon in City*, we experimented a local positioning system based on *Bluetooth* in the *Galata Museum of the Sea* in Genoa, with the aim of supporting the visit of tourists. The system is composed of a set of *Bluetooth*-enabled devices, which we term *base stations* for convenience, scattered around the museum, that constantly listen for new incoming devices. Each base station knows its own position and that of the nearby base stations; a *Bluetooth* device communicates its position to another by using the *Bluetooth Local Positioning Profile*, a protocol that specifies standard data formats and interchange methods for local positioning information. In order to determine the position of a device, we resort to the *triangulation* technique via *lateration*, which uses multiple distance measurements between the

| Landmark | No geolocation | With geolocation |
|---|---|---|
| Basilica S.S. Annunziata | 82.3 | 88.2 |
| Porta dei Vacca | 0 | 93.8 |
| Genoa University Admin. | 50 | 50 |
| Genoa University Art Dept. | 85.7 | 85.7 |
| Palazzo Balbi | 75 | 75 |
| University Library | 50 | 88.9 |
| St. George Palace | 75 | 100 |
| Genoa Cathedral | 80 | 100 |
| Doge's Palace | 100 | 100 |
| Porta Soprana | 100 | 100 |
| Piazza de Ferrari fountain | 100 | 100 |
| Carlo Felice theater | 100 | 100 |
| San Matteo Church | 100 | 100 |
| **Accuracy** | 65.6 | 83.2 |

Table 2. Accuracy of *Agamemnon in City* with and without geolocation.

device and the base stations. The distance between a device and a base station is measured with a technique similar to that described by Feldmann et al. (2003).

### 4.3 Experiments

We tested *Agamemnon in City*, with and without geolocation, in Genoa by simulating a typical visit of a tourist, wandering around the streets of the city and taking photos at 13 selected landmarks to have them recognized by the system. For each landmark we took 100 photos and measured the *success rate* as the number of photos in which the landmark was correctly recognized; the *accuracy* of the system is the *average success rate* over all landmarks. The results, shown in Table 2, where the accuracy is reported in the last row, show that geolocation considerably improves the accuracy of the system, as expected. However, we point out the the poor accuracy of the system without geolocation is due to few monuments, such as "Porta dei Vacca" and "University Library", which seems to suggest that we failed to train the image recognition in a proper way. Moreover, the image recognition of *Agamemnon in City* is the same as the one developed for *Agamemnon*, which was specifically tailored for recognizing monuments in archaeological sites. Finally, while *Agamemnon* has been always tested in ideal conditions, with monuments being clear of obstacles such as crowds, *Agamemnon in City* has been used in a urban environment, where often cars got in the way between the camera and the monument.

## 5. Text entry in handheld devices

*I was afraid of the internet... because I couldn't type* - Jack Welch

While we can still rely on keyboard and mouse to interact with a desktop PC or a laptop, there is a urgent need to find valid text entry tools for handheld devices. Small hardware or software keyboards are only good to write short texts; handwriting recognition systems do not always interpret correctly users' input and generally limit users' writing speed much more than keyboards do, as pointed out by Zhai & Kristensson (2003); speech recognition systems are far from being accurate as well. We note that most today's smartphones are likely to be used to write long texts, such as emails, text documents and spreadsheets, while early mobile

phones were just meant for phone calls and short text messages, for which a small keyboard coupled with *T9* was more than enough.

We now survey some of the text entry tools that have been proposed in recent years, focusing our attention on keyboards and handwriting recognition systems and leaving aside other technologies, such as eye-tracking and speech recognition, that are considerably less popular. At the time we are writing, *Apple* announced the release of the *iPhone 4S*, which features a promising speech recognition system called *Siri*. However, no matter how good a speech recognition system will be, it will never replace a keyboard or a handwriting recognition system, as there are situations where users do not want to annoy people nearby with their voices or let them know their personal matters.

### 5.1 Hardware keyboards

Until recent years mobile phones were almost all equipped with a keypad with 12 keys, each associated to more than one character. Since each key is ambiguous, one needs to tap on a key multiple times to insert a single character, which MacKenzie & Tanaka-Ishii (2007) refer to as the *multitap* method. As a result, the typing speed is very limited, which makes the keypad suitable only to write short messages or notes.

Despite the introduction of *T9* and other text prediction techniques, such as *LetterWise*, presented by MacKenzie et al. (2001), the keypad does not suit the needs of today's handheld devices users. Smartphones such as the *Blackberry* feature a small-size *Qwerty* keyboard, where keys are generally small, which poses serious challenges to people with thick fingers. Some vendors propose portable keyboards whose size is acceptably large, as they are independent of the device, and can be either plugged to the device or connected to it via *Bluetooth*. While portability may not be an issue, as some of them are even foldable, these keyboards can only be used at a desk or in situations where users can comfortably sit. Same considerations apply to infrared laser keyboards.

### 5.2 Virtual keyboards

Virtual or software keyboards are conceived to ensure the portability of handheld devices, while providing a familiar and easy-to use interface. Basically, a *virtual keyboard* is a software that displays the image of a keyboard on the screen which is interacted with by using either a finger or a tiny stylus. The action of pressing a key on a virtual keyboard is usually referred to as a *tap*. Virtual keyboards have the following advantages over their hardware counterparts:

- They do not increase the size of the device.
- Two or more virtual keyboards can be installed in the same device and chosen from by users.
- They can be interacted with by using a stylus, which is independent of the size of users' fingers.

On the downside, virtual keyboards need space on the screen, which is usually already pretty limited, and may not be clearly visible under adverse light conditions. Moreover, hardware keyboards give an immediate tactile feedback to users, which cannot be mimicked in virtual keyboards. Despite this, virtual keyboards are quite popular among handheld devices users. Here we survey some of the most known virtual keyboards that have been proposed over the years. Due to space constraints, the survey is not meant to be exhaustive; for further details we refer the interested reader to Zhai & Kristensson (2003). Without lack of generality, we

assume that the domain of characters is limited to the 26 letters of the English alphabet and the space character, which is the usual approach. Indeed special characters and punctuation marks, except the most used ones such as comma, period, colon and semi-colon, are rarely displayed in a virtual keyboard and a key is usually provided to visualize them upon user's request.

### 5.2.1 Qwerty

Commercial handheld devices provide a *Qwerty* virtual keyboard, as users are usually acquainted with it (Figure 3). However, *Qwerty* is meant to be used with two hands rather than a stylus. Since one needs to move the stylus over the keyboard in order to select characters,

Fig. 3. The *Qwerty* keyboard.

the distance between keys is a critical issue in a virtual keyboard. We remark that *Qwerty* has a landscape orientation, which means that the distance between two keys is typically large. Moreover, letters that form frequent digraphs (i.e. pairs of letters) in English, such as "at" and "in" are distant, which intuitively results in a slower typing speed. Although Baber (1996) claims that the *Qwerty* layout is the result of a random choice, it is common belief that the letters forming frequent digraphs in English have been placed on purpose far from each other to prevent jams in old typewriters, caused by back to back pressures of two close keys. However it is, *Qwerty* was devised well before the advent of handheld devices, thus it is not optimized at all for a pen-based interaction.

### 5.2.2 Fitaly

*Fitaly* (Figure 4) is a virtual keyboard specifically optimized for pen-based text entry [7]. It has been designed by Jean Ichbiah, better known as the inventor of Ada, and today is commercialized by *Textware*, a company founded by Ichbiah himself.

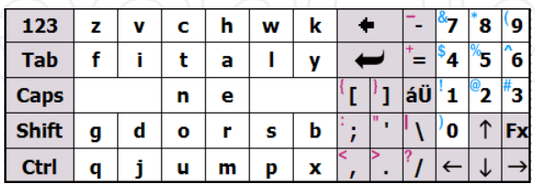


Fig. 4. The *Fitaly* keyboard.

Similarly to *Qwerty*, *Fitaly* owes its name to the concatenation of six adjacent letters in the second row of the keyboard. Unlike *Qwerty*, *Fitaly* has a portrait orientation, which results in a more compact layout. Thus the average distance between two keys is considerably smaller than in *Qwerty*, which results in a minimization of the movement of the hand to jump from

[7] http://www.fitaly.com

one letter to another. Moreover, frequent letters, such as *t*, *a*, *n*, *e*, *o*, *r*, which have a cumulative frequency of 40% according to the tables of Mayzner & Tresselt (1965), are arranged in the middle of the keyboard, while less frequent letters, such as *z* and *x* appear at the edges. As a result, the movements of the stylus are limited in a small area around the centre of the keyboard, and only rarely need to expand to the edges. Finally, two large keys (the ones with no label) are dedicated to the space character; this choice is based on the observation that the space character is much more frequent than any other; in fact, according to the tables of Mayzner & Tresselt (1965), the space has a frequency of 18%, while "e", the second most frequent character, has "only" a frequency of 10%. As of today, *Fitaly* features also a technology known as *Instant Text*, that allow the use of abbreviations to insert words, which further decreases the writing speed.

| | | | | | | |
|---|---|---|---|---|---|---|
| q | f | u | m | c | k | z |
| space | | o | t | h | space | |
| b | s | r | e | a | w | x |
| space | | i | n | d | space | |
| j | p | v | g | l | y | |

Fig. 5. The *Opti* keyboard.

#### 5.2.3 *Opti*

The layout of *Opti*, visualized in Figure 5, is based on the same considerations as *Fitaly*'s, with the difference that *Opti* also accounts for the frequency of English digraphs, as explained by MacKenzie & Zhang (1999). *Opti* has been obtained by first placing the ten most frequent letters in the middle of the keyboard and arranging the other letters based on the most frequent English digraphs. Up to four keys are dedicated to the space character. It is immediate to see that letters forming frequent digraphs such as *th*, *in* and *of* are spatially proximate in the keyboard.

#### 5.2.4 Automatic generation of virtual keyboards

The three keyboards described above have been designed by trial-and-error, which means that different key arrangements have been tried before obtaining a satisfactory one. The optimality of the final layout is only based on the designer judgment. Ideally, one would need to try all possible layouts to pick the best one. However, there are potentially infinite ways of arranging keys in a keyboard, and the keyboard itself may come in shapes other than a rectangle. Obviously, we cannot expect that an algorithm is able to generate all possible layouts of a keyboard; nonetheless, it can try many more than a human in much less time. The generation of a keyboard layout can be modelled as an optimization problem, where typically the values of some parameters can be manually tuned to obtain a layout that meets some predefined constraints.

To the best of our knowledge, the first algorithm that generates a virtual keyboard is described by Getschow et al. (1986). The algorithm sorts the list of the letters of the English alphabet (including the space character) by decreasing frequency and creates three partitions of equal

size, the first containing the top-9 letters and the others containing the remaining. For each partition independently the algorithm tries all possible arrangements of 9 letters in a 3x3 grid and keeps the best. The best layouts of each partition are combined together to form the final keyboard.

The keyboard described by Lewis et al. (1999) is obtained from the analysis of the frequency of the English digraphs. First, a matrix is created containing the relative frequency of unordered English digraphs; the matrix is then used to generate a minimally connected network, with a node for each letter and an edge only between nodes representing letters of digraphs having a frequency above a certain threshold. Each edge of the network has also a weight, representing the strength of the link between two nodes/letters. Finally, a keyboard is created where the distance between two letters connected by an edge with strong weight is minimized.

An original and elegant way to create an optimized virtual keyboard is to resort to physics inspired techniques. The *Hooke* keyboard, proposed by Zhai et al. (2000), is obtained from a mechanical simulation of a mesh of springs, each connecting two characters and tensioned proportionally to the frequency of the digraph formed by the endpoints. The rationale of the approach is that strings connecting two characters that form a frequent digraph need to be pulled together with greater force than two characters in a digraph that rarely occurs. The springs are stretched and then released to obtain the final layout. The layout of *Metropolis* is based on the *Metropolis* algorithm, which is normally used to search for the minimum energy state in a physical system, as described by Binder & Heermann (1988). Since a keyboard must support fast typing, the keys need to be arranged so as to minimize the average time needed to type a word. Suppose that a keyboard is a molecule, the keys are its atoms and the energy of the molecule is the average time to type a word; searching for an optimal layout of the keyboard is tantamount to searching for the arrangements of the atoms that minimize the overall energy of the molecule, which is exactly what the *Metropolis* algorithm is intended for. The average time to type a word can be estimated by using the Fitts (1954) law, as pointed out by Soukoreff & MacKenzie (1995). Fitt's law states that the time required to move from a source to a target area is proportional to the distance between the two areas and inversely proportional to the size of the target area (the smaller is the target area, the more difficult is to hit it). In a virtual keyboard, the time to move the stylus from key $i$ to key $j$ is given by:

$$MT = a + b \log_2 \left( \frac{D_{ij}}{W_j + 1} \right) \tag{1}$$

where $D_{ij}$ is the distance between $i$ and $j$ on the keyboard, $W_j$ is the width of $j$ and $a$, $b$ are two constants whose value is empirically determined, as discussed by Soukoreff & MacKenzie (1995). Usually, $a$ is set to 0. If $P_{ij}$ denotes the frequency of the digraph composed of key $i$ and $j$, the average time in seconds for typing a character in a keyboard is:

$$t = \sum_{i=1}^{27} \sum_{j=1}^{27} b \cdot P_{ij} \left[ \log_2 \left( \frac{D_{ij}}{W_i} + 1 \right) \right] \tag{2}$$

If we assume that English words on average have 5 characters (including space), the number of words that can be written in a minute is $60/5t$.

Finally, we note that recently a 16-year-old high school student, Natalie Nash, created an optimized keyboard layout for people with disabilities by using the simulated annealing algorithm. This story is covered by Krakovsky (2011).

### 5.3 Handwriting recognition systems

Hardware and virtual keyboards are generally considered the fastest way to input text over any other tool, probably due to the fact that people are usually acquainted with them. Ideally, however, it would be nice to input text in a device in the same way as we write on a paper. This explains the flurry of research in handwriting recognition, which is brilliantly surveyed by Plamondon & Srihari (2000) and Tappert & Cha (2007). A handwriting recognition system aims at converting an handwritten text to a digital form, which is a very difficult problem given that humans themselves sometimes cannot interpret the handwriting of somebody else. There are two main research areas: *offline* and *online* recognition. The former refers to the conversion of an handwritten document after the writing is completed, and has numerous applications, such as, but not limited to, reading postal addresses, forms and signatures. The latter refers to the recognition of characters while they are written and is applied in recognition softwares for handheld devices, as users generally prefer to have an immediate feedback of what they write. We here give some details on online recognition and shortly describe softwares that are today used in handheld devices.

#### 5.3.1 How online recognition works

An important concept in online recognition is that of a *stroke*, which refers to any mark made with the pen without lifting it. Information such as the speed of the pen within each stroke, the number of strokes as well as the order and direction of each one are used to considerably improve the accuracy over offline recognition. Online recognition works through three main steps: preprocessing, segmentation and recognition.

The *preprocessing* deals with noise removal, normalization and smoothing. Noise may be due to several reasons, including inaccuracies of the digitization process as well as irregular hand movements; the normalization corrects any irregular slant of the strokes, due to the different handwriting styles of people; finally, smoothing eliminates any blob in the contours of the marks.

*Segmentation* is by far the most challenging problem in handwriting recognition. Usually, a person writes multiple characters without lifting the pen, therefore the recognition must understand which character or group of characters are associated to each stroke. Numerous techniques have been proposed, which we omit here due to space constraints and we refer the interested reader to Liu et al. (2003); Plamondon & Srihari (2000); Tappert & Cha (2007).

Finally, the *recognition* step assigns the segmented marks to characters and/or words. Early recognition systems were based on simple rules, which described the shape of characters based on their geometric features. For instance, a rule that describes the character $x$ may be: "$x$ is two lines that mutually cross and have an inclination of 45 and 135 degree respectively". Such rules are not quite effective to capture the complexity of handwriting and are not used in any commercial products we are aware of. Statistical methods are more efficient, even though they are computationally more expensive and need a lot of training data. Popular techniques are based on artificial neural networks, time-delay neural networks and hidden Markov models. Again, further details can be found in Plamondon & Srihari (2000).

#### 5.3.2 Shorthand writing

Two major obstacles to the development of an ideal handwriting recognition system are the variability in handwriting and segmentation. One way to overcome both is to ask users to

collaborate and adapt their handwriting in order to make the recognition task easier. This is the rationale of *shorthand writing*, pioneered in the pen computing domain by Goldberg & Richardson (1993), who proposed *Unistroke* (Figure 6). *Unistroke* defines a set of symbols, each associated to a character of the alphabet, which are easy to be recognized. To avoid the problem of segmentation, the user is asked to insert each symbol with just one stroke, hence the name *Unistroke*. Compared to a handwriting recognition system using the regular English alphabet, *Unistroke* proved to be twice as faster, as pointed out by Goldberg & Richardson (1993) themselves. However, besides the fact that people may not like the idea of lifting the

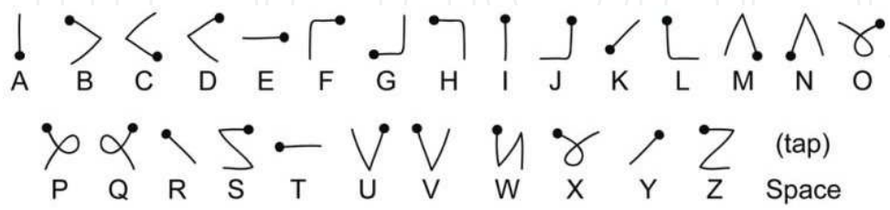



Fig. 6. The *Unistroke* alphabet.

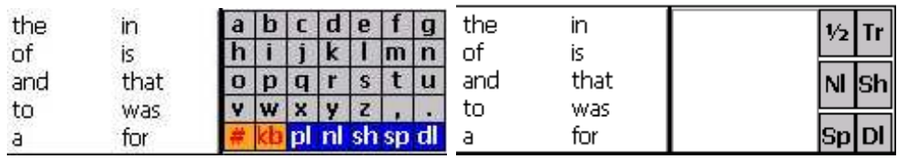
stylus after writing each character, the symbols associated to the alphabet letters need to be learned, which is not always obvious, as most of them do not look like the corresponding character at all. *Unistroke* symbols, indeed, have been devised to be easy to recognize by an algorithm and, more importantly, to be written with just one stroke, rather than to be easy to learn.

*Graffiti*, developed at *Palm Inc.* to run on PDAs based on *Palm OS*, was a major improvement over *Unistroke*, because the symbols have a good visual similarity with the associated characters and consequently can be memorized more easily. However, Fleetwood et al. (2002) found out that *Qwerty* is more efficient than *Graffiti*. First, expert users can write only slightly faster with *Graffiti* than with *Qwerty*, while novice users are considerably slower with *Graffiti*; second, the learning curve is slow, because users take three to six months before writing with *Graffiti* at a speed comparable to that with *Qwerty*; finally, even expert users commit considerably more errors with *Graffiti* than with *Qwerty*.

Other shorthand recognition systems include *Cirrin*, proposed by Mankoff & Abowd (1998), and *Shark*, discussed by Zhai & Kristensson (2003), which associate predetermined gestures to whole words, *Quickwriting*, due to Perlin (1998), which associates strokes to single characters and *Dasher*, described by Ward et al. (2000), which allows writing word with continuous gestures, without lifting the pen at each character or word.

### 5.4 The *WtX* system

*WtX* is a context aware text entry tool for PDAs that we initially developed to support archaeologists taking notes at the excavations, in the context of a project called *RAMSES*, discussed in details by Ancona et al. (1999). The current version runs only on PDAs with the *Windows Mobile* operating system, but we are working on an implementation for both *iPhone* and *iPad*. *WtX* is designed to be a *comfortable*, *easy-to-use* and *flexible* text entry tool, that meets the needs of as many users as possible. *WtX* uses predictive text techniques to enter most of the words by specifying only few letters, hence the comfort; it has a simple interface, that can be used with little or no training at all, hence the ease-of-use; finally, it provides either a virtual keyboard or a handwriting recognition system, based on users' preferences, hence




(a) (b)

Fig. 7. WtX interface with a virtual keyboard (a) and with HRS (b).

the flexibility. Figure 7 shows the interface of *WtX* with a virtual keyboard (a) and with a handwriting recognition system (b).

Basically, *WtX* is composed of two side-by-side areas of equal size; the one on the left side is called *selection area*, and contains up to 10 words loaded from a dictionary, while the one on the right side is called *composition area* and displays either a virtual keyboard or a handwriting recognition system. When a sequence of characters is inserted by using either tool of the composition area, the selection area shows the most frequent words having the sequence as a prefix. Therefore, a word can be inserted by either tapping on each character, if the word is not in the dictionary, or by selecting it when it shows up in the selection area, in which case a space character is automatically appended. We note that at the startup the selection area displays the ten most frequent words in English, that can therefore be selected with just one tap.

Finally, the selection area and the composition area can be swapped upon users' request, so that the former appears on the right side and the latter on the left side of the screen. This option is based on the observation that if the selection area is on the left side left-handed people cover it with their hand, which makes hard to check whether the word being typed shows up. We now describe in greater details some important aspects of *WtX*.

### 5.4.1 The keyboard

The width of the composition area of *WtX* is only half of that normally dedicated to a virtual keyboard, because of the presence of the selection area. Consequently, *Qwerty* is not a good choice for *WtX*, due to its landscape orientation. Initially, we devised a partial keyboard similar to *DotNote* [8], displaying only the most frequent characters in English and one shift key to visualize the others upon request. However, preliminary experiments showed that such a keyboard results in frequent shifts between the two layouts, which considerably decreases the writing speed. Instead, we opted for an alphabetic keyboard, displaying all letters, the space character (the key labelled *sp*) and the two most used punctuation marks (period and comma), as shown in Figure 7 (a).

Although an alphabetic keyboard is not optimized for pen-based entry in the same way as *Opti* or *Fitaly* are, it is still better than *Qwerty*, as discussed by Mackenzie & Soukoreff (2002), and it is easy to learn, as keys are arranged in an intuitive way. The keys in the last row are assigned to special functions. From left to right, the *hash* and *kb* keys are respectively used to display a keypad with numbers and one with special symbols; the *pl* key triggers a function that changes the last inserted word to its plural form; the *nl* key inserts a new line character; the *sh* key activates/deactivates the *FTL* rule, which will be discussed in Section 5.4.4; finally,

[8] *DotNote* is produced by Utilware (`http://www.utilware.com`), a description can be found in Mackenzie & Soukoreff (2002).

the *sp* and *dl* keys are used to insert a space character or delete the last inserted character respectively.

### 5.4.2 The handwriting recognition system

The small size of the composition area severely limits the size of the keyboard, which is likely to make text entry tedious and slow. For this reason we included in *WtX* a handwriting recognition system, based on *Graffiti*, whose interface consists of just one blank box, where users write the characters, and few special keys, as shown in Figure 7 (b). As a result, the composition area does not look as cluttered as with a virtual keyboard. Moreover, an handwriting recognition system, as opposed to a virtual keyboard, is a "heads-up text entry tool", as pointed out by Goldberg & Richardson (1993), meaning that users can write without even looking at the device. Consequently, users' attention must only focus on the selection area. On the downside, *Graffiti* has a low accuracy and frequently misinterprets the characters written by the user.

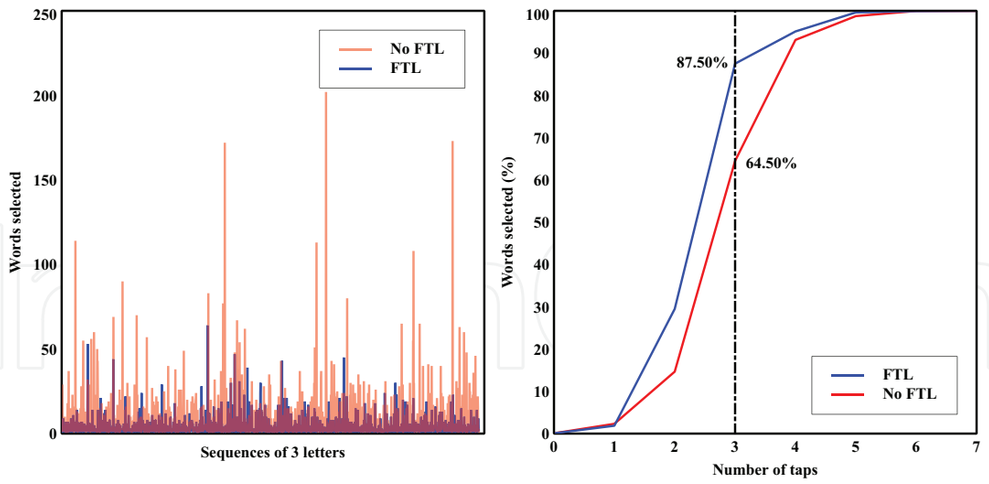
### 5.4.3 The dictionary

In the current version the *WtX* dictionary includes about 14.000 words, each associated with their frequency of use, based on the *British National Corpus*, discussed by Leech et al. (2001). The small size of the dictionary is due to the memory limitations of the devices for which *WtX* was developed. However, in the new releases, notably for *iPhone* and *iPad*, the dictionary will include more words. In main memory the dictionary is represented as a *trie*, which is an ordered rooted tree, where edges are labelled with one letter and each node represents a string, given by the concatenation of the letters on the path from the root to the node. The leaves of the trie represent the words of the dictionary.

### 5.4.4 The *FTL* rule

The goal of text prediction is the minimization of the number of taps or strokes needed to insert a word. In the *WtX* dictionary the average word length, weighted by word frequency, is 4.4 characters; words show up in the selection area after an average of 2.43 taps or strokes. In order to further decrease the average number of taps/strokes, we propose what we term the *First-Third-Last* (*FTL*) rule, which asks users to write a word by specifying its first, third and last character before specifying the others, starting from the second one, in the order in which they appear. *FTL* is based on the observation that any three-letter sequence is likely to match a considerably higher number of words of the dictionary if the characters of the sequence are interpreted as their prefix than if they are interpreted as their first, third and last ones. This fact is illustrated in Figure 8(a), which plots every possible three-letter sequence against the number of words of the dictionary matched by the sequence with (blue line) and without (transparent red line) the *FTL* rule.

In other words, the *FTL* rule acts as a powerful filter on the dictionary, by selecting, for any given three-letter sequence, a small number of words, thus improving the likelihood that a word shows up in the selection area after only three taps, no matter how long the word is. As shown in Figure 8(b), up to 87.50% of the words of the dictionary show up in the selection area after only three taps if the *FTL* rule is used; this number drops to 64.50% if the *FTL* rule is not used.




(a) (b)

Fig. 8. The effect of using the FTL rule.

We note that we need on average 2.32 taps/strokes to insert a word of the dictionary with the *FTL* rule, which is only slightly better than the 2.43 taps/strokes needed without it. This is certainly due to the fact that the average number of taps/strokes is weighted by the frequency of the words and the *FTL* rule is not effective on frequent words, because they are usually very short. Indeed, the average length of the 100 most used English words, which have a 50% cumulative frequency, is 3.35 characters, which is considerably less than the non-weighted average of all English words (5 characters).

*FTL* is similar in spirit to abbreviation systems such as *InstantText* of Fitaly, where a word is inserted by specifying its first letter and one or more non necessarily consecutive letters in the order in which they appear. For instance, the word "dichlorodifluoromethane" can be inserted as "didime", "dcfm", or "doooh". However, once a character is inserted, the system has no clue to know its position within the word, which is likely to make the selection of the words from the dictionary more ambiguous than with *FTL*. We may also argue that from a user standpoint it might be easier to know that there is only one way to write a word, as in the case of *FTL*, instead of countless abbreviations, as in *InstantText*. However, we cannot draw any conclusion at this point, as we have not compared yet *FTL* against *InstantText*, although it is part of our future work.

### 5.4.5 Context-awareness

One important aspect of *WtX* is that we designed it to be context aware, which, to the best of our knowledge, is a novelty for a text entry tool. More specifically, *WtX* can handle multiple dictionaries and select the words from the one that is more appropriate to the current user's context. Consider the example of a doctor who uses *WtX* to either send emails to his friends or fill in the medical record of one of her patients. While in the first case a general English dictionary suits perfectly her needs, in the second case a dictionary of medical terms is much more appropriate, because it contains words that may not be found in a general purpose dictionary and it does not contains words that are unlikely to be used in a medical record. The second aspect is particularly important, as ideally the selection area of *WtX* should suggest only words that are appropriate to a given context. Context awareness in

| Session | Speed | Accuracy | Comfort |
|---|---|---|---|
| *Qwerty* | **13.33** | 0.43 | 0.02 |
| *WtX* + Keyboard | **9.64** | 4.55 | **0.03** |
| *WtX* + *FTL* + *Graffiti* | 6.43 | 3.57 | **0.03** |

Table 3. *WtX* experiment results.

*WtX* is realized through an Application Programming Interface (API) that allows third-party applications to communicate to *WtX* any change of the context and how the dictionary should change accordingly. We experimented the *WtX* context-awareness in *WardInHand*, where the dictionary automatically changes according to the position of the doctor within an hospital; this provides the doctor with a dictionary of terms appropriate to a specific ward, without the need of manually selecting it.

#### 5.4.6 Experiments

A text entry tool is typically evaluated by asking a group of people to type a sample text and measuring two parameters, namely *speed* and *accuracy*. The first is usually measured as number of words per minute averaged on all users, while the second as the inverse of the number of misspelled words. We introduce *comfort* as a third parameter, which we define as the tiredness perceived by the user while writing a long text.

The measurement of *comfort* is complicated, because, as opposed to speed and accuracy, it is a subjective parameter. Speed alone can not be a reliable measure of comfort. Indeed, the average speed of a user may be high because s/he inserts quickly the very first sentences of the sample text and slows down on the subsequent sentences because of tiredness. The experiments described by Ancona & Quercini (2009) showed indeed that there is a correlation between speed variation over time and comfort; the higher the variation, the lower the comfort. Therefore, we measure comfort as the inverse of the average speed variation over the time taken to write the sample text.

In our experiments we selected 10 participants and we asked them to write a sample text with 454 words by using, in the order, *Qwerty*, *WtX* with the virtual keyboard and *WtX* with *Graffiti* and *FTL*. More details on the settings of the experiments can be found in Ancona & Quercini (2009). The results are shown in Table 3. As expected, participants wrote more quickly when using *Qwerty*, as they were familiar with it. However, accuracy is low and speed variation is high, which means a low level of comfort. With *WtX*, accuracy remarkably improves, as most of the words are directly selected from the dictionary, and comfort benefits from it. Despite the high number of misspelled words due to *Graffiti*, users reported to feel less tired when using *WtX* with *FTL* and *Graffiti*. Although we are planning to further investigate this point in the future, we are convinced that the *FTL* rule plays an important role in improving comfort, due to the minimization of the average number of taps needed to insert words.

## 6. Concluding remarks

In this chapter we covered two important aspects of the interaction with handheld devices, namely *context-awareness* and *text entry*. We analysed the key challenges in the design and implementation of context-aware applications, with a particular focus on our past research projects, namely *WardInHand*, *Agamemnon* and *Past*. We surveyed the most popular text entry tools, especially virtual keyboards and handwriting recognition systems, and described *WtX*, a context-aware text entry tool for PDAs that we developed few years ago.

The recent advent and success of sophisticated and economic smartphones, such as the *iPhone*, calls for a revitalization of our past research projects, that have been developed when the technology was not mature enough to fully support all their innovative ideas. Sensors, for instance, extremely simplify the implementation of context-aware applications, as we illustrated in several examples throughout this chapter. Moreover, natural user interfaces, such as speech recognition softwares, have been improved a lot in the last decade and start to appear in commercial products, such as the *iPhone 4S*, released in October 2011. Finally, smartphones' computational power has been increasing at a high pace, which paves the way to applications that were virtually impossible only a decade ago.

## 7. References


Abowd, G. D. & Mynatt, E. D. (2000). Charting Past, Present, and Future Research in Ubiquitous Computing, *ACM Trans. Comput.-Hum. Interact.* 7: 29–58.

Ancona, M., Cappello, M., Casamassima, M., Cazzola, W., Conte, D., Pittore, M., Quercini, G., Scagliola, N. & Villa, M. (2006). Mobile Vision and Cultural Heritage: the AGAMEMNON Project, *in* B. Schiele, L. Paletta & L. V. Gool (eds), *First International Workshop on Mobile Vision*, pp. 3–17.

Ancona, M., Conte, D., Quercini, G. & Casamassima, M. (2007). Attention-Aware Cultural Heritage Applications on Mobile Phones, *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, IEEE, pp. 1–8.

Ancona, M., Coscia, E., Rubattino, C. & Megliola, M. (2003). Horizontal Versus Vertical Development of the HCI in the Ward-in-Hand Project, *4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine*, IEEE, pp. 27–30.

Ancona, M., Dodero, G. & Gianuzzi, V. (1999). RAMSES: a Mobile Computing System for Field Archaeology, *in* Hans-W.Gellersen (ed.), *Handheld and Ubiquitous Computing*, Vol. 1707 of *Lecture Notes in Computer Science*, Springer Verlag, Heidelberg, pp. 222–233.

Ancona, M., Dodero, G., Gianuzzi, V., Bocchini, O., Vezzoso, A., Traverso, A. & Antonacci, E. (2000). Exploiting Wireless Networks for Virtual Archaeology: the PAST Project, *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*.

Ancona, M., Dodero, G., Minuto, F., Guida, M. & Gianuzzi, V. (2000). Mobile Computing in a Hospital: the WARD-IN-HAND Project, *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 2*, SAC '00, ACM, New York, NY, USA, pp. 554–556.

Ancona, M., Locati, S. & Romagnoli, A. (2001). Context and Location Aware Textual Data Input, *Proceedings of the 2001 ACM Symposium on Applied Computing*, SAC '01, ACM, New York, NY, USA, pp. 425–428.

Ancona, M. & Quercini, G. (2009). Text Entry in PDAs with WtX, *The Ergonomics Open Journal* 2: 185–195.

Azuma, R. T. (1997). A Survey of Augmented Reality, *Presence* 6: 355–385.

Baber, C. (1996). *Beyond the Desktop: Designing and Using Interaction Devices*, Academic Press.

Binder, K. & Heermann, D. (1988). *Monte Carlo Simulation in Statistical Physics*, SpringerVerlag.

Bricon-Soufand, N. & Newman, C. R. (2007). Context Awareness in Health Care: a Review, *International Journal of Medical Informatics* 76: 2 – 12.

Feldmann, S., Kyamakya, K., Zapater, A. & Lue, Z. (2003). An Indoor Bluetooth-Based Positioning System: Concept, Implementation and Experimental Evaluation, *International Conference on Wireless Networks*, ICWN '03, CSREA Press, pp. 109–113.

Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement, *Journal of Experimental Psychology* pp. 381 – 391.

Fleetwood, M., Byrne, M., Centgraf, P., Dudziak, K., Lin, B. & Mogilev, D. (2002). An Analysis of Text-Entry in Palm OS: Graffiti and the Virtual Keyboard, *Proc. of the 46th Human Factors and Ergonomics Society Annual Meeting*, pp. 617–621.

Getschow, C. O., Rosen, M. J. & Goodenough-Trepagnier (1986). A systematic Approach to Design a Minimum Distance Alphabetical Keyboard, *Proceedings of RESNA (Rehabilitation Engineering Society of North America) 9th Annual Conference*, pp. 396–398.

Goldberg, D. & Richardson, C. (1993). Touch-typing with a Stylus, *Proceedings of the INTERCHI '93 conference on Human factors in computing systems*, INTERCHI '93, IOS Press, Amsterdam, The Netherlands, pp. 80–87.

Istepanian, R., Laxminarayan, S. & Pattichis, C. S. (eds) (2005). *M-Health: Emerging Mobile Health Systems*, Springer.

Krakovsky, M. (2011). Success at 16, *Commun. ACM* 54: 20–20.

Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T. & Campbell, A. T. (2010). A Survey of Mobile Phone Sensing, *Comm. Mag.* 48: 140–150.

Leech, G., Rayson, P. & Wilson, A. (2001). *Word Frequencies in Written and Spoken English: Based on the British National Corpus*, Longman, London.

Lewis, J. R., Kennedy, P. J. & LaLomia, M. J. (1999). Development of a Digram-based Typing Key Layout for Single-Finger/Stylus Input, *Human Factors and Ergonomics Society 43rd Annual Meeting.*

Liu, Z.-Q., Cai, J. & Buse, R. (2003). *Handwriting Recognition: Soft Computing and Probabilistic Approaches*, Springer.

MacKenzie, I. S., Kober, H., Smith, D., Jones, T. & Skepner, E. (2001). LetterWise: Prefix-Based Disambiguation for Mobile Text Input, *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, ACM, New York, NY, USA, pp. 111–120.

MacKenzie, I. S. & Zhang, S. X. (1999). The Design and Evaluation of a High-Performance Soft Keyboard, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI is the Limit*, CHI '99, ACM, New York, NY, USA, pp. 25–31.

MacKenzie, I. & Tanaka-Ishii, K. (2007). *Text Entry Systems*, Morgan Kaufmann.

Mackenzie, S. I. & Soukoreff, W. R. (2002). Text Entry for Mobile Computing: Models and Methods,Theory and Practice, *Human-Computer Interaction* 17(2 & 3): 147–198.

Mankoff, J. & Abowd, G. D. (1998). Cirrin: a Word-level Unistroke Keyboard for Pen Input, *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, ACM, New York, NY, USA, pp. 213–214.

Mayzner, M. S. & Tresselt, M. E. (1965). Tables of Single-Letter and Digram Frequency Counts for Various Word-Length and Letter-Position Combinations, *Psychonomic Monograph Supplements* 1(2): 12 – 32.

Oviatt, S. & Cohen, P. (2000). Perceptual User Interfaces: Multimodal Interfaces that Process What Comes Naturally, *Commun. ACM* 43: 45–53.

Perlin, K. (1998). Quikwriting: Continuous Stylus-Based Text Entry, *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*, UIST '98, ACM, New York, NY, USA, pp. 215–216.

Pittore, M., Cappello, M., Ancona, M. & Scagliola, N. (2005). Role of Image Recognition in Defining the User's in 3G Phone Applications: the AGAMEMNON Experience, *IEEE International Conference on Image Processing*, ICIP 2005, pp. 1012 – 1015.

Plamondon, R. & Srihari, S. N. (2000). On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey, *IEEE Trans. Pattern Anal. Mach. Intell.* 22: 63–84.

Raman, T. & Chen, C. (2008). Eyes-Free User Interfaces, *Computer* 41(10): 100 –101.

Schilit, B. N., Adams, N. & Want, R. (1994). Context-Aware Computing Applications, *Workshop on Mobile Computing Systems and Applications*, WMCSA '94, IEEE, pp. 89–101.

Soukoreff, R. W. & MacKenzie, I. S. (1995). Theoretical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard, *Behaviour & Information Technology* 14: 370–379.

Tappert, C. C. & Cha, S.-H. (2007). English Language Handwriting Recognition Interfaces, *in* I. S. MacKenzie & K. Tanaka-Ishii (eds), *Text Entry Systems: Mobility, Accessibility, Universality*, Morgan Kaufmann, chapter 6, pp. 123–137.

Turk, M. (2004). Computer Vision in the Interface, *Commun. ACM* 47: 60–67.

Turk, M. & Robertson, G. (2000). Perceptual User Interfaces (Introduction), *Commun. ACM* 43: 32–34.

Vertegaal, R. (2003). Attentive User Interfaces, *Commun. ACM* 46: 31–33.

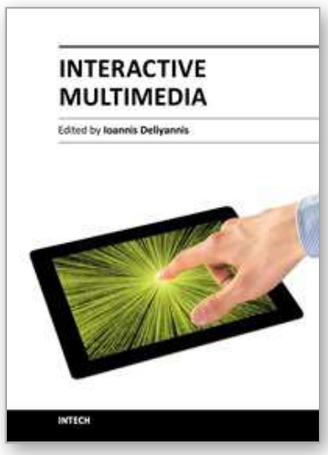
Wang, J., Zhai, S. & Canny, J. (2006). Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study, *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, ACM, New York, NY, USA, pp. 101–110.

Ward, D. J., Blackwell, A. F. & MacKay, D. J. C. (2000). Dasher - A Data Entry Interface Using Continuous Gestures and Language Models, *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, ACM, New York, NY, USA, pp. 129–137.

Warrior, J., McHenry, E. & McGee, K. (2003). They Know Where You Are, *IEEE Spectrum* 40: 20–25.

Zhai, S., Hunter, M. & Smith, B. A. (2000). The Metropolis Keyboard - An Exploration of Quantitative Techniques for Virtual Keyboard Design, *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, ACM, New York, NY, USA, pp. 119–128.

Zhai, S. & Kristensson, P.-O. (2003). Shorthand Writing on Stylus Keyboard, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM, New York, NY, USA, pp. 97–104.




**Interactive Multimedia**

Edited by Dr Ioannis Deliyannis

ISBN 978-953-51-0224-3
Hard cover, 312 pages
**Publisher** InTech
**Published online** 07, March, 2012
**Published in print edition** March, 2012

Interactive multimedia is clearly a field of fundamental research, social, educational and economical importance, as it combines multiple disciplines for the development of multimedia systems that are capable to sense the environment and dynamically process, edit, adjust or generate new content. For this purpose, ideas, theories, methodologies and inventions are combined in order to form novel applications and systems. This book presents novel scientific research, proven methodologies and interdisciplinary case studies that exhibit advances under Interfaces and Interaction, Interactive Multimedia Learning, Teaching and Competence Diagnosis Systems, Interactive TV, Film and Multimedia Production and Video Processing. The chapters selected for this volume offer new perspectives in terms of strategies, tested practices and solutions that, beyond describing the state-of-the-art, may be utilised as a solid basis for the development of new interactive systems and applications.



INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen